Read Tanenbaum, Bos: Chapter 10.4, 11.5.

Exercises

1. Suppose a memory allocator has a freelist with blocks, in order, of sizes 20, 8, 40, 36, 14, 18, 24, and 30 bytes. A request comes for a 24 byte block, then a 20 byte block, and then an 18 byte block.

    (a) Which blocks would be chosen using a first fit approach?

    (b) Which blocks would be chosen using a best fit approach?

    (c) Which blocks would be chosen using a worst fit approach?

2. Explain the difference between internal and external fragmentation.

3. Consider a Linux system with 4K pages and 32-bit addresses. A program is loaded so that its code resides at virtual address $600000. Part of the page table for the process is shown below.

    (a) What physical address does the logical address $6021A8 map to?

    (b) What physical address does the logical address $6040C3 map to?

    (c) Explain what will happen if the program references location $600000.

Page	Frame	Present Bit
600	–	0
601	001A0	1
602	001A1	1
603	001A2	1
604	21002	1
605	–	0
606	7FFE2	1

4. A computer with 32-bit addresses uses a two-level page table. Virtual addresses are split into a 9-bit top level page table field, and 11-bit second level page table field, and an offset. How large are the pages and how many of them are there in the address space?

5. A machine has 48-bit virtual addresses and 32-bit physical addresses. Pages are 8Kb. How many entries are needed for the page table?

6. A typical `emacs` editor process uses 2Mb of program text, 6Mb of data, and 1Mb of stack. Suppose three users are running emacs. Approximately how much physical memory are they using in total?

7. In Linux, the heap and stack are paged to a paging file kept in a special disk or partition, but the program code is not. Why?

8. Suppose `data` is a large two-dimensional array.
   Why will these two loops run at significantly different speeds?

```
// version 1                          // version 2
for (i = 0; i<SIZE; i++)             for (j = 0; j<SIZE; j++)
  for (j = 0; j<SIZE; j++)             for (i = 0; i<SIZE; i++)
    data[i][j] = value;                  data[i][j] = value;
```